Paradigmas de Programación

Compilación
Inferencia de tipos

1er cuatrimestre de 2024

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Introducción

Inferencia de tipos

Compiladores

¿Qué es un compilador?

Un compilador es un programa que traduce programas:

Entrada:	programa escrito en un **lenguaje fuente**.
Salida:	programa escrito en un **lenguaje objeto**.

Este proceso de traducción debe **preservar la semántica**.
(O, mejor: aquellos aspectos que nos interesen de la semántica).

Compiladores

¿Para qué queremos un compilador?

Motivación principal

Traducir de lenguajes de alto nivel a lenguajes de bajo nivel.

$$x = \frac{-b+\sqrt{b^2-4ac}}{2a}$$

```
movsd xmm0, QWORD PTR -32[rbp]
mulsd xmm0, xmm0
movsd xmm2, QWORD PTR -24[rbp]
movsd xmm1, QWORD PTR .LC0[rip]
mulsd xmm1, xmm2
mulsd xmm1, QWORD PTR -16[rbp]
subsd xmm0, xmm1
call sqrt@PLT
subsd xmm0, QWORD PTR -32[rbp]
movsd xmm1, QWORD PTR .LC1[rip]
divsd xmm0, xmm1
movsd xmm1, QWORD PTR -24[rbp]
mulsd xmm0, xmm1
movsd QWORD PTR -8[rbp], xmm0
```

Compiladores

Fases típicas de un compilador

	programa fuente
ANÁLISIS SINTÁCTICO	$\downarrow$
	árbol sintáctico
ANÁLISIS SEMÁNTICO	$\downarrow$
	árbol sintáctico con anotaciones
COMPILACIÓN	$\downarrow$
	representación intermedia
OPTIMIZACIÓN	$\downarrow$
	representación intermedia optimizada
GENERACIÓN DE CÓDIGO	$\downarrow$
	programa objeto

Introducción

Inferencia de tipos

Inferencia de tipos

Notación

Términos **sin** anotaciones de tipos:

$$U ::= x \mid \lambda x.\, U \mid U\, U \mid \mathsf{True} \mid \mathsf{False} \mid \mathsf{if}\ U\ \mathsf{then}\ U\ \mathsf{else}\ U$$

Términos **con** anotaciones de tipos:

$$M ::= x \mid \lambda x : \tau.\, M \mid M\, M \mid \mathsf{True} \mid \mathsf{False} \mid \mathsf{if}\ M\ \mathsf{then}\ M\ \mathsf{else}\ M$$

Notamos $\mathtt{erase}(M)$ al término sin anotaciones de tipos que resulta de borrar las anotaciones de tipos de M.

Ejemplo: $\mathtt{erase}((\lambda x : \mathsf{Bool}.\, x)\, \mathsf{True}) = (\lambda x.\, x)\, \mathsf{True}$.

Inferencia de tipos

Definición

Un término U sin anotaciones de tipos es **tipable** sii existen:

un contexto de tipado Γ

un término con anotaciones de tipos M

un tipo τ

tales que $\texttt{erase}(M) = U$ y $\Gamma \vdash M : \tau$.

El **problema de inferencia de tipos** consiste en:

- Dado un término U, determinar si es tipable.
- En caso de que U sea tipable:
  hallar un contexto Γ, un término M y un tipo τ
  tales que $\texttt{erase}(M) = U$ y $\Gamma \vdash M : \tau$.

Veremos un algoritmo para resolver este problema.

Inferencia de tipos

El algoritmo se basa en manipular tipos *parcialmente conocidos*.

Ejemplo — tipos parcialmente conocidos

- En x True sabemos que $x : \text{Bool} \to \text{X}_1$.
- En if $x\,y$ then True else False sabemos que $x : \text{X}_2 \to \text{Bool}$.

Incorporamos *incógnitas* $(\text{X}_1, \text{X}_2, \text{X}_3, \ldots)$ a los tipos.

Vamos a necesitar resolver *ecuaciones* entre tipos con incógnitas.

Ejemplo — ecuaciones entre tipos

- $(\text{X}_1 \to \text{Bool}) \stackrel{?}{=} ((\text{Bool} \to \text{Bool}) \to \text{X}_2)$
  tiene solución: $\text{X}_1 := (\text{Bool} \to \text{Bool})$ y $\text{X}_2 := \text{Bool}$.
- $(\text{X}_1 \to \text{X}_1) \stackrel{?}{=} ((\text{Bool} \to \text{Bool}) \to \text{X}_2)$
  tiene solución: $\text{X}_1 := (\text{Bool} \to \text{Bool})$ y $\text{X}_2 := (\text{Bool} \to \text{Bool})$.
- $(\text{X}_1 \to \text{Bool}) \stackrel{?}{=} \text{X}_1$
  no tiene solución.

Unificación

Suponemos fijado un conjunto finito de constructores de tipos:

- Tipos constantes: Bool, Int, $\dots$.
- Constructores unarios: (List $\bullet$), (Maybe $\bullet$), $\dots$.
- Constructores binarios: $(\bullet \to \bullet)$, $(\bullet \times \bullet)$, (Either $\bullet$ $\bullet$), $\dots$.
- (Etcétera).

Los tipos se forman usando incógnitas y constructores:

$$\tau ::= \mathsf{X}_n \mid C(\tau_1, \dots, \tau_n)$$

La **unificación** es el problema de resolver sistemas de ecuaciones entre tipos con incógnitas.

Veremos primero un algoritmo de unificación.
Luego lo usaremos para dar un algoritmo de inferencia de tipos.

Unificación

Una **sustitución** es una función que a cada incógnita le asocia un tipo.

Notamos:

$$\{X_{k_1} := \tau_1, \ldots, X_{k_n} := \tau_n\}$$

a la sustitución $\mathbf{S}$ tal que $\mathbf{S}(X_{k_i}) = \tau_i$ para cada $1 \leq i \leq n$ y $\mathbf{S}(X_k) = X_k$ para cualquier otra incógnita.

Si τ es un tipo, escribimos $\mathbf{S}(\tau)$ para el resultado de reemplazar cada incógnita de τ por el valor que le otorga $\mathbf{S}$.

Ejemplo — aplicación de una sustitución a un tipo

Si $\mathbf{S} = \{X_1 := \mathrm{Bool},\ X_3 := (X_2 \to X_2)\}$, entonces:

$$\mathbf{S}((X_1 \to \mathrm{Bool}) \to X_3) = ((\mathrm{Bool} \to \mathrm{Bool}) \to (X_2 \to X_2))$$

Unificación

Un **problema de unificación** es un conjunto finito E de ecuaciones entre tipos que pueden involucrar incógnitas:

$$E = \{\tau_1 \stackrel{?}{=} \sigma_1, \tau_2 \stackrel{?}{=} \sigma_2, \ldots, \tau_n \stackrel{?}{=} \sigma_n\}$$

Un **unificador** para E es una sustitución $\mathbf{S}$ tal que:

$$\mathbf{S}(\tau_1) = \mathbf{S}(\sigma_1)$$

$$\mathbf{S}(\tau_2) = \mathbf{S}(\sigma_2)$$

$$\ldots$$

$$\mathbf{S}(\tau_n) = \mathbf{S}(\sigma_n)$$

Unificación

En general, la solución a un problema de unificación no es única.

Ejemplo — problema de unificación con infinitas soluciones

$$\{X_1 \stackrel{?}{=} X_2\}$$

tiene infinitos unificadores:

- $\{X_1 := X_2\}$
- $\{X_2 := X_1\}$
- $\{X_1 := X_3,\ X_2 := X_3\}$
- $\{X_1 := \mathsf{Bool},\ X_2 := \mathsf{Bool}\}$
- $\{X_1 := (\mathsf{Bool} \to \mathsf{Bool}),\ X_2 := (\mathsf{Bool} \to \mathsf{Bool})\}$
- $\ldots$

Unificación

Una sustitución $\mathbf{S}_A$ es **más general** que una sustitución $\mathbf{S}_B$ si existe una sustitución $\mathbf{S}_C$ tal que:

$$\mathbf{S}_B = \mathbf{S}_C \circ \mathbf{S}_A$$

es decir, $\mathbf{S}_B$ se obtiene instanciando variables de $\mathbf{S}_A$.

Para el siguiente problema de unificación:

$$E = \{(\mathsf{X}_1 \to \mathsf{Bool}) \stackrel{?}{=} \mathsf{X}_2\}$$

las siguientes sustituciones son unificadores:

- $\mathbf{S}_1 = \{\mathsf{X}_1 := \mathsf{Bool},\ \mathsf{X}_2 := (\mathsf{Bool} \to \mathsf{Bool})\}$
- $\mathbf{S}_2 = \{\mathsf{X}_1 := \mathsf{Int},\ \mathsf{X}_2 := (\mathsf{Int} \to \mathsf{Bool})\}$
- $\mathbf{S}_3 = \{\mathsf{X}_1 := \mathsf{X}_3,\ \mathsf{X}_2 := (\mathsf{X}_3 \to \mathsf{Bool})\}$
- $\mathbf{S}_4 = \{\mathsf{X}_2 := (\mathsf{X}_1 \to \mathsf{Bool})\}$

¿Qué relación hay entre ellas? (¿Cuál es más general que cuál?).

Algoritmo de unificación de Martelli–Montanari

Dado un problema de unificación E (conjunto de ecuaciones):

- Mientras $E \neq \varnothing$, se aplica sucesivamente alguna de las seis reglas que se detallan más adelante.
- La regla puede resultar en una `falla`.
- De lo contrario, la regla es de la forma $E \to_{\mathbf{S}} E'$.
  La resolución del problema E se reduce a resolver otro problema E', aplicando la sustitución $\mathbf{S}$.

Hay dos posibilidades:

1. $E = E_0 \to_{\mathbf{S}_1} E_1 \to_{\mathbf{S}_2} E_2 \to \ldots \to_{\mathbf{S}_n} E_n \to_{\mathbf{S}_{n+1}} \texttt{falla}$
   En tal caso el problema de unificación E no tiene solución.
2. $E = E_0 \to_{\mathbf{S}_1} E_1 \to_{\mathbf{S}_2} E_2 \to \ldots \to_{\mathbf{S}_n} E_n = \varnothing$
   En tal caso el problema de unificación E tiene solución.

Algoritmo de unificación de Martelli–Montanari

$$\{X_n \stackrel{?}{=} X_n\} \cup E \quad \xrightarrow{\texttt{Delete}} \quad E$$

$$\{C(\tau_1, \dots, \tau_n) \stackrel{?}{=} C(\sigma_1, \dots, \sigma_n)\} \cup E \quad \xrightarrow{\texttt{Decompose}} \quad \{\tau_1 \stackrel{?}{=} \sigma_1, \dots, \tau_n \stackrel{?}{=} \sigma_n\} \cup E$$

$$\{\tau \stackrel{?}{=} X_n\} \cup E \quad \xrightarrow{\texttt{Swap}} \quad \{X_n \stackrel{?}{=} \tau\} \cup E$$

si τ no es una incógnita

$$\{X_n \stackrel{?}{=} \tau\} \cup E \quad \xrightarrow{\texttt{Elim}}_{\{X_n := \tau\}} \quad E' = \{X_n := \tau\}(E)$$

si X_n no ocurre en τ

$$\{C(\tau_1, \dots, \tau_n) \stackrel{?}{=} C'(\sigma_1, \dots, \sigma_m)\} \cup E \quad \xrightarrow{\texttt{Clash}} \quad \texttt{falla}$$

si $C \neq C'$

$$\{X_n \stackrel{?}{=} \tau\} \cup E \quad \xrightarrow{\texttt{Occurs-Check}} \quad \texttt{falla}$$

si $X_n \neq \tau$
y X_n ocurre en τ

Algoritmo de unificación de Martelli–Montanari

Teorema (Corrección del algoritmo de Martelli–Montanari)

1. El algoritmo termina para cualquier problema de unificación E.
2. Si E no tiene solución, el algoritmo llega a una `falla`.
3. Si E tiene solución, el algoritmo llega a $\varnothing$:

   $$E = E_0 \rightarrow_{\mathbf{S}_1} E_1 \rightarrow_{\mathbf{S}_2} E_2 \rightarrow \ldots \rightarrow_{\mathbf{S}_n} E_n = \varnothing$$

   Además, $\mathbf{S} = \mathbf{S}_n \circ \ldots \circ \mathbf{S}_2 \circ \mathbf{S}_1$ es un unificador para E.

   Además, dicho unificador es el *más general* posible.
   (Salvo renombre de incógnitas).

Definición (Unificador más general)

Notamos $\mathrm{mgu}(E)$ al unificador más general de E, si existe.

Algoritmo de unificación de Martelli–Montanari

Ejemplo

Calcular unificadores más generales para los siguientes problemas de unificación:

- $\{(X_2 \to (X_1 \to X_1)) \stackrel{?}{=} ((\mathsf{Bool} \to \mathsf{Bool}) \to (X_1 \to X_2))\}$
- $\{X_1 \stackrel{?}{=} (X_2 \to X_2),\ X_2 \stackrel{?}{=} (X_1 \to X_1)\}$

Algoritmo $\mathbb{W}$ de inferencia de tipos

El algoritmo $\mathbb{W}$ recibe un término U sin anotaciones de tipos.

Procede recursivamente sobre la estructura de U:

- Puede fallar, indicando que U no es tipable.
- Puede tener éxito.
  En tal caso devuelve una tripla (Γ, M, τ),
  donde $\mathtt{erase}(M) = U$ y $\Gamma \vdash M : \tau$ es válido.

Escribimos $\mathbb{W}(U) \rightsquigarrow \Gamma \vdash M : \tau$ para indicar que el algoritmo de inferencia tiene éxito cuando se le pasa U como entrada y devuelve una tripla (Γ, M, τ).

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\frac{}{\mathbb{W}(\mathsf{True}) \;\rightsquigarrow\; \varnothing \vdash \mathsf{True} : \mathsf{Bool}}$$

$$\frac{}{\mathbb{W}(\mathsf{False}) \;\rightsquigarrow\; \varnothing \vdash \mathsf{False} : \mathsf{Bool}}$$

$$\frac{\mathsf{X}_k \text{ es una incógnita fresca}}{\mathbb{W}(x) \;\rightsquigarrow\; x : \mathsf{X}_k \vdash x : \mathsf{X}_k}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\begin{array}{rcl}\mathbb{W}(U_1) & \rightsquigarrow & \Gamma_1 \vdash M_1 : \tau_1 \\ \mathbb{W}(U_2) & \rightsquigarrow & \Gamma_2 \vdash M_2 : \tau_2 \\ \mathbb{W}(U_3) & \rightsquigarrow & \Gamma_3 \vdash M_3 : \tau_3\end{array}$$

$$\frac{\mathbf{S} = \text{mgu}\left(\begin{array}{l}\{\tau_1 \stackrel{?}{=} \text{Bool}, \tau_2 \stackrel{?}{=} \tau_3\} \cup \\ \{\Gamma_i(x) \stackrel{?}{=} \Gamma_j(x) \mid i, j \in \{1,2,3\},\ x \in \mathit{dom}(\Gamma_i) \cap \mathit{dom}(\Gamma_j)\}\end{array}\right)}{\begin{array}{rcl}\mathbb{W}(\text{if } U_1 \text{ then } U_2 \text{ else } U_3) & \rightsquigarrow & \mathbf{S}(\Gamma_1) \cup \mathbf{S}(\Gamma_2) \cup \mathbf{S}(\Gamma_3) \vdash \\ & & \mathbf{S}(\text{if } M_1 \text{ then } M_2 \text{ else } M_3) : \mathbf{S}(\tau_2)\end{array}}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\dfrac{\begin{array}{c} \mathbb{W}(U) \ \rightsquigarrow \ \Gamma_1 \vdash M : \tau \\ \mathbb{W}(V) \ \rightsquigarrow \ \Gamma_2 \vdash N : \sigma \\ \mathsf{X}_k \text{ es una incógnita fresca} \\ \mathbf{S} = \mathrm{mgu}\{\tau \stackrel{?}{=} \sigma \to \mathsf{X}_k\} \cup \{\Gamma_1(x) \stackrel{?}{=} \Gamma_2(x) : x \in \Gamma_1 \cap \Gamma_2\} \end{array}}{\mathbb{W}(U\,V) \ \rightsquigarrow \ \mathbf{S}(\Gamma_1) \cup \mathbf{S}(\Gamma_2) \vdash \mathbf{S}(M\,N) : \mathbf{S}(\mathsf{X}_k)}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

$$\frac{\mathbb{W}(U) \;\rightsquigarrow\; \Gamma \vdash M : \tau \qquad \sigma = \begin{cases} \Gamma(x) & \text{si } x \in \Gamma \\ \text{una incógnita fresca } \mathsf{X}_k & \text{si no} \end{cases}}{\mathbb{W}(\lambda x.\, U) \;\rightsquigarrow\; \Gamma \ominus \{x\} \vdash \lambda x : \sigma.\ M : \sigma \to \tau}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

Teorema (Corrección del algoritmo $\mathbb{W}$)

1. Si U no es tipable, $\mathbb{W}(U)$ falla al resolver alguna unificación.
2. Si U es tipable, $\mathbb{W}(U) \ \rightsquigarrow\ \Gamma \vdash M : \tau$,
   donde $\texttt{erase}(M) = U$ y $\Gamma \vdash M : \tau$ es un juicio válido.

   Además, $\Gamma \vdash M : \tau$ es el juicio de tipado más general posible.
   Más precisamente, si $\Gamma' \vdash M' : \tau'$ es un juicio válido y $\texttt{erase}(M') = U$, existe una sustitución $\mathbf{S}$ tal que:

$$\begin{array}{rcl} \Gamma' & \supseteq & \mathbf{S}(\Gamma) \\ M' & = & \mathbf{S}(M) \\ \tau' & = & \mathbf{S}(\tau) \end{array}$$

Algoritmo $\mathbb{W}$ de inferencia de tipos

Ejercicio. Aplicar el algoritmo de inferencia sobre los siguientes términos:

- $\lambda x.\, \lambda y.\, y\, x$
- $(\lambda x.\, x\, x)(\lambda x.\, x\, x)$

Lecturas sugeridas

- Benjamin C. Pierce. Types and Programming Languages. Capítulo 22 (Type reconstruction)
- M. H. Sorensen and P. Urzyczyn. Lectures on the Curry-Howard Isomorphism (Sección 3.2)

¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ? ? ? ? ? ? ? ? ?